Descomposición QR

Por Marcelo Gabriel Ulrich
Aprendizaje Automatico - Septiembre 2025
UNSAM

Repaso

Antes de arrancar, vamos con un repaso (rápido) de los temas necesarios para entender la descomposición QR.

Regresión Lineal

Tenemos $X^T_{original} = (X_1, X_2, \ldots, X_p)$ y el vector $Y = (y_1, y_2, \ldots, y_n)$.
La regresión lineal asume

$$\hat{Y} = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

$$\hat{Y} = X^T \beta$$

$$\text{con } X^T = [1 | X^T_{original}]$$

Por lo tanto, buscamos estimar β que minimice una función de costos/error. En general usamos Minimos Cuadrados, que busca minimizar

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - \hat{y}_i(\beta))^2$$

obteniendo así el Estimador de Cuadrados Minimos:

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

El tema es que...

...en la practica, calcular la inversa de X^TX presenta un problemón:
Si X tiene multicolinealidad perfecta (es decir, hay un X_i linealmente dependiente de las otras X), el determinante de la matriz X^TX es cero. Eso rompe la formula de estimadores de beta (no se puede calcular la inversa), por lo que hay que arremangarse y hacer otras cosas (ejemplo, selección de variables con SVD/PCA, etc.).
Pero que pasa en casos menos extremos, cuando estos vectores no son linealmente dependientes, pero son muy parecidos, es decir, con multicolinealidad imperfecta (alta correlación pero no 1)? El determinante se acerca mucho a cero, pero no es estrictamente cero.
Otra forma de justificarlo es que la matriz X^TX "sufre" de la cuadratura del numero de condición:

- El numero de condición, $cond(A) = ||A||||A^{-1}||$, indica que tan sensible es la matriz a los errores/pequeñas perturbaciones en los datos de entrada.
- Si el numero es cercano a 1, todo bien, mientras más grande, más sensible a las perturbaciones en los datos

Esto genera un problema a la hora de calcular la inversa: un pequeño error de redondeo (por aritmetica de punto flotante) genera grandes cambios en el resultado. Es decir, el calculo de $(X^TX)^{-1}$ se vuelve numericamente inestable, lo que hace que la estimación de los betas se vuelva poco confiable.
[EJEMPLO COMPUTACIONAL]
La multicolinealidad es un problema muy frecuente en *la naturaleza* (todo conjunto de datos no generado artificialmente), por lo que hace que la formula de estimación de los betas sea poco practica (no da soluciones confiables).

Y ahora quien podrá ayudarnos?
[insertar meme]
Yo! La descomposición de matrices!

Descomposición de matrices

Al "desarmar" $X^T X$ en otras matrices de manera inteligente, podemos aprovechar ciertas propiedades de las matrices para aligerar los calculos y generar soluciones más estables, más rápidas, etc.
Ya vimos SVD, en la que se descompone X en 3 matrices: dos ortogonales y una diagonal con valores singulare. La SVD es super completa y nos dá mucha info sobre la matriz que descompone. Sin embargo, en el contexto de la regresión lineal, es información redundante, por lo que podriamos aspirar a encontrar otras tecnicas de descomposición que prioricen otras cosas por sobre la información extra, como la estabilidad y/o tiempo de computo.
Hoy les traigo otro metodo de descomposición de matrices muy usado, que promete ser más eficiente computacionalmente y más directo que SVD... La Descomposición QR.

Descomposición QR

Así cómo en Singular Value Decomposition $X = UDV^T$, este metodo propone descomponer X en otras matrices. En particular

$$X = QR$$

[MEME de OBVIO]
La matriz Q es de $N \times P + 1$ y ortogonal, es decir, sus columnas son linealmente independientes/ortogonales/"perpendiculares" y sus normas son 1.
La matriz R es de $P + 1 \times P + 1$, triangular superior y contiene coeficientes (ya vamos a ver qué son...).

Descomposición QR: Ultilidad

La gracia es que al reemplazar en la formula del Estimador de Cuadrados Minimos, genera lo siguiente:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\hat{\beta} = ((\mathbf{QR})^T\mathbf{QR})^{-1}(\mathbf{QR})^T\mathbf{y}$$

$$\hat{\beta} = (\mathbf{R}^T\mathbf{Q}^T\mathbf{QR})^{-1}\mathbf{R}^T\mathbf{Q}^T\mathbf{y}$$

Descomposición QR: Ultilidad

$$\hat{\beta} = (\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R})^{-1}\mathbf{R}^T\mathbf{Q}^T\mathbf{y}$$

Como Q es ortogonal, vale que $Q^TQ = QQ^T = I$, entonces:

$$\hat{\beta} = (\mathbf{R}^T\mathbf{I}\mathbf{R})^{-1}\mathbf{R}^T\mathbf{Q}^T\mathbf{y}$$

$$\hat{\beta} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{Q}^T\mathbf{y}$$

$$\hat{\beta} = \mathbf{R}^{-1}(\mathbf{R}^T)^{-1}\mathbf{R}^T\mathbf{Q}^T\mathbf{y}$$

$$\hat{\beta} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{y}$$

Descomposición QR: Ultilidad

La gracia es que R^{-1} es más estable numericamente que $(X^T X)^{-1}$. El calculo de la inversa de R es más "resistente" a errores de redondeo generados por la aritmetica de punto flotante del calculo computacional. Es decir, calcular R^{-1} arrastra menos error por redondeo que calcular $(X^T X)^{-1}$, llegando a soluciones más confiables/estables.
Pero... Por qué? Qué son Q y R?

Descomposición QR: Generar la matriz Q

Q es una matriz cuyas columnas q_i son ortogonales entre sí, lo que vimos que es clave para simplificar las cuentas.
Pero, cómo la generamos?
Usando el algoritmo de Gram-Schmidt para ortogonalizar vectores!

Algoritmo de Gram-Schmidt

En el libro de Elements of Statistical Learning, se describe de la siguiente manera:
Dado un conjunto de vectores columna de X, $\{x_0, x_1, \ldots, x_p\}$

1. $z_0 = x_0 = 1$
2. $\forall j = 1, 2, \ldots, p$:

$$\gamma_{l,j} = \frac{\langle z_l, x_j \rangle}{\langle z_l, z_l \rangle} \quad \forall l = 0, 1, \ldots, j-1$$

$$z_j = x_j - \sum_{k=0}^{j-1} \gamma_{k,j} z_k$$

Entonces, la base de vectores residuos, $\{z_0, z_1, \ldots, z_p\}$, es una base ortogonal.

[meme] Oye, despacio cerebrito...

Qué son estos residuos? bueno, vamos paso a paso.

1. Tenemos el vector $z_0 = x_0 = [1, 1, \ldots, 1]^T$
2. Quiero generar un vector z_1 tome al vector x_1 y lo "independice" de x_0. Para eso, proyectamos x_1 en x_0 (para ver "cuanto apunta" x_1 en la dirección de x_0) y le resto a x_1 esa proyección. Así, me quedo con la componente de x_1 que es perpendicular a x_0, o sea, linealmente independiente.
   2.1. Como sabemos por definición, la proyección del vector x_1 sobre x_0 es $\hat{x}_{0,1} = \frac{\langle x_0, x_1 \rangle}{||x_0||^2} x_0 = \frac{\langle z_0, x_1 \rangle}{||z_0||^2} z_0$
   2.2. Entonces, obtenemos $z_1 = x_1 - \hat{x}_{0,1}$

[Animación de la proyección ortogonal]

Ahora, si quiero encontrar z_2, no solo voy a tener que "independizarla" de x_0/z_0, sino que tambien de z_1.
Entonces $z_2 = x_2 - \hat{x}_{0,2} - \hat{x}_{1,2}$
Con $\hat{x}_{0,2} = \frac{\langle z_0, x_2 \rangle}{||z_0||^2} z_0$ y $\hat{x}_{1,2} = \frac{\langle z_1, x_2 \rangle}{||z_1||^2} z_1$
[animación?]
Generalizando para los p vectores columna, obtenemos las expresiones vistas antes

$$\gamma_{l,j} = \frac{\langle z_l, x_j \rangle}{\langle z_l, z_l \rangle} = \frac{\langle z_l, x_j \rangle}{||z_l||^2} \quad \forall l = 0, 1, \ldots, j-1$$

$$z_j = x_j - \sum_{l=0}^{j-1} \gamma_{l,j} z_l$$

Entonces, genero dos matrices:

1. Z, con los residuos z como columnas
2. Γ, una matriz triangular con los coeficientes γ y el resto 0 (recordar que calculamos $\gamma_{0,1}$ y no $\gamma_{1,0}$)

Y obtengo $X = Z\Gamma$

Validando

Por definición: $z_j = x_j - \sum_{l=0}^{j-1} \gamma_{l,j} z_l$, por lo que es lo mismo que

$$x_j = z_j + \sum_{l=0}^{j-1} \gamma_{l,j} z_l$$

Para el caso x_0, $x_0 = z_0 = [z_0][1]^T$
Para el caso x_1, $x_1 = z_1 + \gamma_{0,1} z_0 = [z_0\ z_1][\gamma_{0,1}\ 1]^T$
Para el caso x_2, $x_2 = z_2 + \gamma_{0,2} z_0 + \gamma_{0,1} z_1 = [z_0\ z_1\ z_2][\gamma_{0,2}\ \gamma_{1,2}\ 1]^T$
En notación matricial vale

$$X = [\, z_0 \quad z_1 \quad z_2 \,] \begin{bmatrix} 1 & \gamma_{0,1} & \gamma_{0,2} \\ 0 & 1 & \gamma_{1,2} \\ 0 & 0 & 1 \end{bmatrix}$$

Ahhh, o sea que Q son los vectores columna z y R son los coeficientes $\gamma_{l,j}$?

No pero casi, sino se llamaría descomposición $Z\Gamma$ xddd
Estamos casi ahí, el unico problema es que Z no es una matriz ortogonal, ya que sus columnas no tienen norma 1 (nunca se lo pedimos a las z...)

Para lograrlo, necesitamos dividir las componentes de Z por sus normas. Cómo? Fácil, un poco de magia (o sea, algebra lineal)

Definimos una matriz diagonal D, en la cual $d_{i,i} = ||z_i||$, por lo que D^{-1} tiene los reciprocos de D, $d^*_{i,i} = \frac{1}{||z_i||}$

Si multiplicamos Z y D^{-1}, obtenemos los vectores z_i normalizados, es decir:

$$q_i = \frac{z_i}{||z_i||}$$

Por lo que definimos

$$Q = [q_0\ q_1\ \dots\ q_p]$$

Cómo D es diagonal, vale que $D^{-1}D = I$, por lo que nos lo podemos sacar del [REDACTED] la nada
Entonces nos queda

$$X = ZI\Gamma = (ZD^{-1})(D\Gamma)$$

$$X = QR$$

Entonces, que hay en Q y R?

Q tiene columnas ortonormales generadas a partir de las columnas de X a traves del algoritmo de Gram-Schmidt
R tiene en su diagonal principal las normas de los vectores z_i, es decir, de los residuos de ortogonalizar las columnas de X; y en el resto de su triangulo superior, los "terminos cruzados" de nuestras columnas x_i originales, extraidos a partir de proyecciones ortogonales.

Corre o no corre?

[Armar codigo para probar casos de matrices rectangulares, ortogonales perfectas, casi ortogonales, multicolineales y casi multicolineales]

Muchas gracias por su atención!

Preguntas?